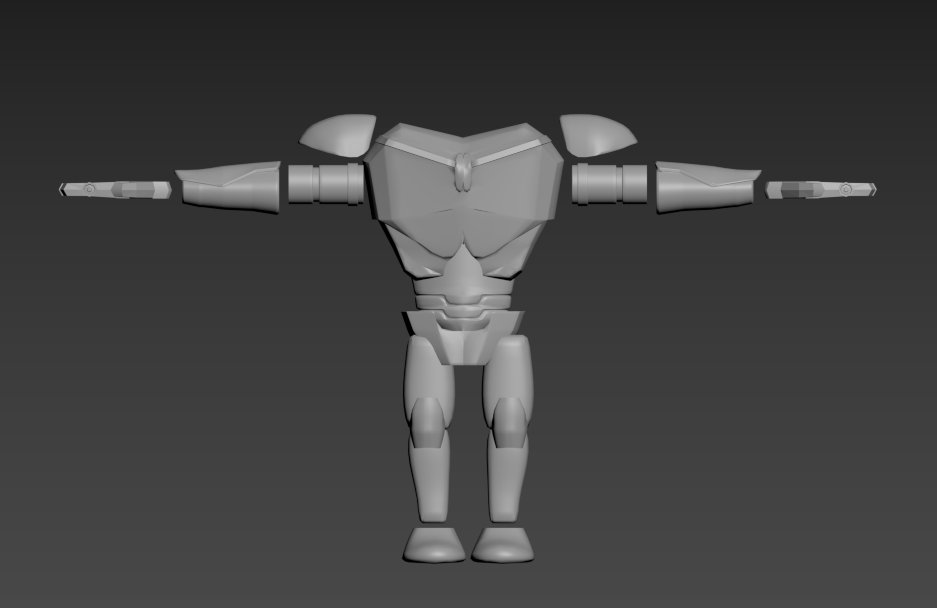
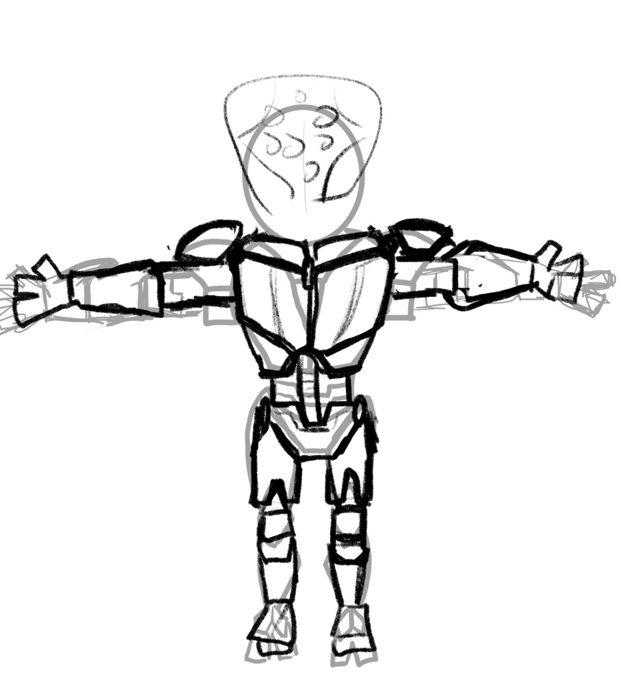
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **주차** | 6주차 | **기간** | 10.24 - 10.30 | **지도교수** | (서명) | |
| 이번주 한일 요약 | 모델링 3 몸통 제작  유데미 강의 수강 | | | | |

<상세 수행내용>

이번주에는 강의를 수강 하고 3번째 캐릭터의 몸통 모델링을 완료 하였습니다.



이번주에는 적 캐릭터의 AI를 c++코드를 이용해 만들어 보았습니다.

AI 컨트롤러를 생성하고 AI객체가 조준하는 방법, 이동 방법, 시선에 따른 판단을 하는 방법을 배웠으며, Behavior tree와 블랙보드 사용법과 그 안에 있는 다양한 기능, 이를 C++로 키 설정 하는 법을 배웠습니다.

**◎ AI 컨트롤러 생성하고 설정**

c++프로젝트 생성 - AIController

(ShooterAIController)

상속받는 BP\_ShooterAIController 생성

BP\_ShooterCharacter에서 pawn의 AI Controll class를 BP\_ShooterAIController로 설정

**◎ AI로 조준하기**

== ShooterAIController.h

class SIMPLESHOOTER\_API AShooterAIController : public AAIController

{

GENERATED\_BODY()

protected:

virtual void BeginPlay() override;

};

== ShooterAIController.cpp

#include "ShooterAIController.h"

#include "Kismet/GameplayStatics.h"

void AShooterAIController::BeginPlay()

{

Super::BeginPlay();

APawn\* PlayerPawn = UGameplayStatics::GetPlayerPawn(GetWorld(), 0);

SetFocus(PlayerPawn);

}

**◎ 내비매시를 이용해서 AI 이동하기**

내비매시 : AI가 이동할 수 있는 공간

왼쪽에서 Nav mesh Bound V를 이용해 생성 (p로 보이게)

== ShooterAIController.h

class SIMPLESHOOTER\_API AShooterAIController : public AAIController

{

GENERATED\_BODY()

public:

virtual void Tick(float DeltaSeconds) override;

protected:

virtual void BeginPlay() override;

};

== ShooterAIController.cpp

void AShooterAIController::Tick(float DeltaSeconds)

{

Super::Tick(DeltaSeconds);

APawn \*PlayerPawn = UGameplayStatics::GetPlayerPawn(GetWorld(), 0);

MoveToActor(PlayerPawn, 200);

}

**◎ AI 시선 확인**

== ShooterAIController.h

private:

UPROPERTY(EditAnywhere)

float AcceptanceRadius = 200;

== ShooterAIController.cpp

void AShooterAIController::BeginPlay()

{

Super::BeginPlay();

}

void AShooterAIController::Tick(float DeltaSeconds)

{

Super::Tick(DeltaSeconds);

APawn \*PlayerPawn = UGameplayStatics::GetPlayerPawn(GetWorld(), 0);

if (LineOfSightTo(PlayerPawn))

{

SetFocus(PlayerPawn);

MoveToActor(PlayerPawn, AcceptanceRadius);

}

else

{

ClearFocus(EAIFocusPriority::Gameplay);

StopMovement();

}

}

**◎ 행동트리 와 블랙보드**

Behavior tree 생성

Black Board 생성

== ShooterAIController.h

private:

UPROPERTY(EditAnywhere)

class UBehaviorTree\* AIBehavior;

== ShooterAIController.cpp

void AShooterAIController::BeginPlay()

{

Super::BeginPlay();

if (AIBehavior != nullptr)

{

RunBehaviorTree(AIBehavior);

}

}

BP\_ShooterAIController 들어와서 AIBehavior BT\_EnemyAI로 설정

**◎ C++로 블랙보드 키 설정**

BB\_EnemyAI : (vector) PlayerLoctaion 생성

#include "BehaviorTree/BlackboardComponent.h"

void AShooterAIController::BeginPlay()

{

Super::BeginPlay();

if (AIBehavior != nullptr)

{

RunBehaviorTree(AIBehavior);

APawn\* PlayerPawn = UGameplayStatics::GetPlayerPawn(GetWorld(), 0);

GetBlackboardComponent()->SetValueAsVector(TEXT("PlayerLocation"), PlayerPawn->GetActorLocation());

}

}

**◎ 행동 트리**

BT 들어와서 시퀀스 밑에 move to / wait / move to / wait 연결

moveto를 플레이어로케이션으로 설정

BB에서 startLoction 만들기

두번째 move to StartLocation

void AShooterAIController::BeginPlay()

{

Super::BeginPlay();

if (AIBehavior != nullptr)

{

RunBehaviorTree(AIBehavior);

APawn\* PlayerPawn = UGameplayStatics::GetPlayerPawn(GetWorld(), 0);

GetBlackboardComponent()->SetValueAsVector(TEXT("PlayerLocation"), PlayerPawn->GetActorLocation());

GetBlackboardComponent()->SetValueAsVector(TEXT("StartLocation"), GetPawn()->GetActorLocation());

}

}

**◎ BT 데코레이터 및 셀렉터**

BT에 Selector생성 - 시퀀스(chase / investigate) 연결

chase에 add decorator - 블랙보드

BB에 LastKnownPlayerLocation 생성

블랙보드 Chase에서 isset을 PlayerLocation으로

그리고 moveto연결 (PlayerLocation)

investigate도 moveto(LastKnownPlayerLocation)

블랙보드노드에 observe aborts를 both로 설정

**◎ C++로 커스텀 BTTask 구성하기**

investigate도 blackboard 입히고 오른쪽에 Move to 추가

C++로 BTTask\_BlackboradBase 상속 파일 생성

== Source/SimpleShooter/SimpleShooter.Build.cs

public class SimpleShooter : ModuleRules

{

public SimpleShooter(ReadOnlyTargetRules Target) : base(Target)

{

PCHUsage = PCHUsageMode.UseExplicitOrSharedPCHs;

PublicDependencyModuleNames.AddRange(new string[] { "Core", "CoreUObject", "Engine", "InputCore" });

PublicDependencyModuleNames.AddRange(new string[] { "Core", "CoreUObject", "Engine", "InputCore", "GameplayTasks" });

PrivateDependencyModuleNames.AddRange(new string[] { });

== h

#pragma once

#include "CoreMinimal.h"

#include "BehaviorTree/Tasks/BTTask\_BlackboardBase.h"

#include "BTTask\_ClearBlackboardValue.generated.h"

UCLASS()

class SIMPLESHOOTER\_API UBTTask\_ClearBlackboardValue : public UBTTask\_BlackboardBase

{

GENERATED\_BODY()

public:

UBTTask\_ClearBlackboardValue();

};

== cpp

#include "BTTask\_ClearBlackboardValue.h"

UBTTask\_ClearBlackboardValue::UBTTask\_ClearBlackboardValue()

{

NodeName = "Clear Blackboard Value";

}

BT에서 마지막에 Clear BlackBoard Value 노드 생성 후 연결

안에서 LastKnown- 설정

wait 설정

|  |  |  |  |
| --- | --- | --- | --- |
| **문제점 정리** |  | | |
| **해결방안** |  | | |
| **다음주차** |  | **다음기간** |  |
| **다음주 할일** | 유데미 강의 마무리  졸업작품 캐릭터4 얼굴 모델링 | | |
| **지도 교수**  **Comment** |  | | |